REPORT DOCUMENTATION PAGE

Form Approved
OMB NO. 0704-0188

1. AGENCY USE ONLY (Leave Blank)	2. REPORT DATE	3. REPORT TYPE AND DATES COVERED
	January 6, 2004	Interim Progress-29 Jul 2003 – 08 Aug 2003

4. TITLE AND SUBTITLE

Bell Labs Algorithms Pow Wow

5. FUNDING NUMBERS

N00014-03-M-0141

6. AUTHOR(S)

F. Bruce Shepherd, Chandra Chekuri, Anupam Gupta, Vahab Mirrokni, Hadas Shachnai, Seffi Naor, Gordon Wilfong, Mansoor Alicherry, George Karakostas, Adrian Vetta, Dan Bienstock, Randeep Bhatia

7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES)

Lucent Technologies Bell Laboratories
600 Mountain Avenue
Murray Hill, NJ 07974

8. PERFORMING ORGANIZATION REPORT NUMBER

9. SPONSORING / MONITORING AGENCY NAME(S) AND ADDRESS(ES)

Office of Naval Research
Ballston Tower One
800 North Quincy Street
Arlington, VA 22217 – 5660

10. SPONSORING / MONITORING AGENCY REPORT NUMBER

11. SUPPLEMENTARY NOTES

12 a. DISTRIBUTION / AVAILABILITY STATEMENT

Approved for public release; distribution is unlimited.

12 b. DISTRIBUTION CODE

A

13. ABSTRACT (Maximum 200 words)

For two weeks researchers in algorithms gathered to remind each other of some old unsolved problems and to present some new ones. People broke off into groups according to their interest in certain problems. Problems were presented in the following areas: the k-cut and multiway cut, Min-max programming, triangle finding in linear time, directed multicut, minimum power *k*-connected subgraph, job scheduling with communication delays, bounded-degree biclique cover, metric labeling, priority Steiner Tree, network design: orientation constraints, edge-coloring dynamic bipartite multi-graphs, edge coloring bipartite multi-hypergraphs, optimal cost chromatic partition (OCCP), single –source unsplittable flow, confluent flow, combinatorial algorithms for short-path-decomposable flows, shortest path routing, and packing dijoins and feedback arc sets. In each of these areas the problems were identified and progress made during the meeting was presented.

14. SUBJECT TERMS

15. NUMBER OF PAGES
21

16. PRICE CODE

17. SECURITY CLASSIFICATION OR REPORT	18. SECURITY CLASSIFICATION ON THIS PAGE	19. SECURITY CLASSIFICATION OF ABSTRACT	20. LIMITATION OF ABSTRACT
UNCLASSIFIED	**UNCLASSIFIED**	**UNCLASSIFIED**	**UNLIMITED**

Bell Labs Algorithms Pow Wow: July 29 - August 8, 2003

THE WORD "POW WOW" comes from the Algonquin word "PauWau" which was used to describe the medicine men and spiritual leaders. Early Europeans thought the word referred to an entire event. As Indian tribes learned English, they accepted this definition. Powwow time has thus come to mean people meeting together, to join in dancing, singing, visiting, renewing old friendships and making new ones. This is a time to renew thoughts of the old ways and to preserve a rich heritage.

For two weeks researchers in algorithms gathered to remind each other of some old unsolved problems and to present some new ones. People broke off somewhat randomly into groups according to when and whether they became interested in certain of the problems. Here is a list of the problems presented, as well as some of the progress that was made during the meeting. We shall see later whether some of the groups end up settling any of the questions, or other results emerging from this discourse. We gratefully acknowledge the support received from the Office of Naval Research by way of a basic research grant.

Participants:

Mansoor Alicherry, Bell Labs, mansoor@research.bell-labs.com
Matthew Andrews, Bell Labs, andrews@research.bell-labs.com
Elliot Anshelevich, Cornell University, anshelev@research.bell-labs.com
Randeep Bhatia, Bell Labs, randeep@research.bell-labs.com
Dan Bienstock, Columbia University, dano@ieor.columbia.edu
Chandra Chekuri, Bell Labs, chekuri@research.bell-labs.com
Ken Clarkson, Bell Labs, clarkson@research.bell-labs.com
Steve Fortune, Bell Labs, sjf@research.bell-labs.com
Anupam Gupta, Carnegie Mellon University, anupamg@cs.cmu.edu
George Karakostas, McMaster University, karakos@mcmaster.ca
Vahab Mirrokni, MIT, mirrokni@theory.lcs.mit.edu
Seffi Naor, Technion University, naor@cs.Technion.AC.IL
Hadas Shachnai, Bell Labs, hadas@research.bell-labs.com
Bruce Shepherd, Bell Labs, bshep@research.bell-labs.com
Adrian Vetta, McGill University, vetta@jeff.cs.mcgill.ca
Gordon Wilfong, Bell Labs, gtw@research.bell-labs.com
Lisa Zhang, Bell Labs, ylz@research.bell-labs.com

Chandra Chekuri

0.1 Some Simple Cut Problems

Two fundamental graph partitioning problems are the k-cut problem and the multiway cut problem. In both problems we are given an undirected edge-weighted graph $G = (V, E)$ with $w(e)$ denoting the weight of edge $e \in E$. In the k-cut problem the goal is to find a minimum weight set of edges to separate the graph into at least k disconnected components. In the multiway cut problem we are given a set of k terminals, $X \subseteq V$, and the goal is to find a minimum weight set of edges to separate the graph into components, such that each terminal is in a different connected component. Chekuri, Guha, and Naor [3] defined a common generalization of the two problems that they refer to as the Steiner k-cut problem. We are given an undirected weighted graph G, a set of *terminals* $X \subseteq V$, and an integer $k \leq |X|$. The goal is to find a minimum weight cut that separates the graph into k components with vertex sets $V_1, V_2, \ldots, V_k$, such that $V_i \cap X \neq \emptyset$ for $1 \leq i \leq k$. If $X = V$, we obtain the k-cut problem. If $|X| = k$ we obtain the multiway cut problem.

The k-cut problem can be solved in polynomial time for fixed k [8, 9], but is NP-hard when k is part of the input [8]. In contrast, the multiway cut problem is NP-hard for all $k \geq 3$ [4]. It follows that the Steiner k-cut problem is NP-hard for all $k \geq 3$. For the multiway cut problem Calinescu, Karloff and Rabani [2] gave a $1.5 - 1/k$ approximation using an interesting geometric relaxation. Karger et al. [10] improved the analysis of the integrality gap of this relaxation and obtained an approximation ratio of $1.3438 - \epsilon_k$, where ϵ_k tends to 0 as k tends to ∞. For the k-cut problem Saran and Vazirani [13] gave a $2 - \frac{2}{k}$ approximation algorithm using a greedy algorithm. Recently, two different 2-approximations for the k-cut problem were obtained. The algorithm of Naor and Rabani [11] is based on rounding a linear programming formulation of the problem and the algorithm of Ravi and Sinha [12] is based on the notion of network strength.

In [3], two approximation algorithms are presented for the Steiner k-cut problem. The first algorithm is combinatorial and achieves a factor of $2 - \frac{2}{k}$. The algorithm is based on choosing cuts from the Gomory-Hu tree and it is very similar to approximation algorithms developed for the k-cut problem and the multiway cut problem [14]. The second algorithm is a 2-approximation algorithm for the Steiner k-cut problem which is based on rounding a linear programming formulation. Although the formulation is a generalization of the formulation in [11] (for the k-cut problem), the rounding scheme differs substantially. The rounding in [11] exploits the properties of the optimal solution to the LP relaxation. These properties do not hold for the relaxation of the Steiner k-cut problem. The new rounding is based on the primal dual algorithm and analysis of Goemans and Williamson [7] for the Steiner tree problem. As a consequence, the rounding algorithm extends to any feasible solution of the linear programming formulation.

In [3], a bi-directed formulation for the global minimum cut problem is also presented and it is shown that that the linear relaxation of this formulation is exact.

Open Problems:

- Is the k-cut problem Max-SNP hard?
- Is there a better than 2 approximation for the k-cut problem?
- Is there a better than a 2 approximation for the Steiner k-cut problem?
- The integrality gap of the LP in [11, 3] is 2 even for $k = 2$. Is there a strengthening of the LP that gives an optimal solution for $k = 2$? This was accomplished by Dan Bienstock [1] by using lifting. Can we use his ideas to obtain an improved integrality gap for larger values of k.
- For the multiway cut problem, Freund and Karloff [6] have shown that the integrality gap of the LP in [2] is at least $8/(7+1/(k-1))$. However the best upper bound known is $1.3438 - \epsilon_k$ [10]. Can this gap be bridged?
- During the workshop Dan Bienstock queried whether the polyhedron of vectors x such that $x(H) \geq 2$ for each spanning 2-connected subgraphs gave a gap of better than 2. Shepherd and Vetta thus asked whether there is an $\alpha > 1$ every $4k$-connected graph could be decomposed into αk spanning 2-edge-connected subgraphs. Naor and Shepherd noted that if this were true, one could also find a better-than-2 approximation for the min-cost 2-edge-connected subgraph problem (an open problem).

Anupam Gupta

0.2 Min-Max Programming

We are given a directed complete bipartite graph $G = (U, V, E)$, where $E = (U \times V) \cup (V \times U)$, with edge weights $d : E \to \mathbf{Z}$. A *feasible labeling* of the vertices is an assignment of values $x : U \cup V \to \mathbf{Z}$ which satisfies

$$x(u) \geq \max_{v \in V} \{x(v) + d(v,u)\} \qquad \forall u \in U; x(v) \geq \min_{u \in U} \{x(u) + d(u,v)\} \qquad \forall v \in V. \tag{1}$$

Problem: Decide in polynomial time whether G has a feasible labeling.

Previous Results: The problem is known to be in NP $\cap$ co-NP, and can be solved in pseudo-polynomial poly$(n, d_{\max})$ time. If the edge weights are non-negative, polynomial-time solutions are known. These results can be found, e.g., in a paper of Moehring, Skutella and Stork (SIAM J. Comput., 2002).

0.3 Triangle finding in linear time

Problem: Given a undirected graph $G = (V, E)$, decide in linear time whether G has a triangle (i.e., an induced K_3).

Previous Results: There is a trivial solution in $O(n^3)$ time, which can be improved to $O(n^\omega)$ by fast matrix multiplication. An algorithm running in $O(m^{1.4})$ time can be found in a paper of Alon,Yuster and Zwick (STOC, 1994), along with references to above results.

0.4 Directed Multicut

Given a directed graph $G = (V, A)$, and a set of tuples $(s_i, t_i) \in V \times V$ for $1 \leq i \leq k$, the *minimum directed multicut* problem asks for a set $S \subseteq A$ of minimum cardinality such that $G - S$ has no s_i-t_i path for all i.

Problem: Does there exist an $O(\log n)$ approximation?

Previous Results: An $O(\min\{k, \sqrt{n}\})$ approximation is known due to Gupta (SODA, 2003), extending results of Cheriyan, Karloff and Rabani (FOCS, 2001). The natural linear program has an integrality gap of $(k - \epsilon)$, shown by Saks Samrodnitsky and Zosin (2002).

Vahab Mirrokni

1. **Minimum Power k-connected subgraph:** Given a graph $G(V, E)$ with $w : E(G) \rightarrow R$, the power of vertex $v \in V(G)$ is $P(v) = \max_{vu \in E(G)} w(vu)$. The power of graph G is the sum of the powers of vertices, i.e, $P(G) = \sum v \in V(G) P(v)$. Our goal is to find a k-(edge-)connected subgraph of a given graph, G, with the minimum total power, minimize$_{G'}$ subgraph $_G P(G')$ where G' is k-(edge-)connected.

   Comments: The problem is NP-complete even for $k = 1$. $O(k)$ approximations are known for edge-connected and vertex connected subgraph. We are seeking approximation algorithm with $o(k)$ ratios.

2. **Job Scheduling with Communication Delays:** We are given n unit length jobs, m parallel machines, and a precedence constraint graph $G(V, E)$. We are also given a communication delay ρ. $(u, v) \in E(G)$ means that if jobs u and v are scheduled in the same machine, then v's starting time should be greater than or equal to u's completion time ($s_v \geq s_u + 1$). If u and v are scheduled in different machines v's starting time should be greater than or equal to u's completion time plus ρ ($s_v \geq s_u + 1 + \rho$). The goal is to schedule all jobs and minimize the total makespan.

   Comments: The problem is NP-complete. There are two cases for the problem: duplication of job is allowed or not. Notice duplicating a job may help to decrease makespan sometimes. If duplication is allowed then an algorithm of $O(\frac{\log \rho}{\log \log \rho})$ is known. Getting constant factor approximations or inapproximability results for both cases are desirable.

Hadas Shachnai

0.5 The Bounded-degree Biclique Cover Problem

Let $B = (V_1, V_2, E)$ be a bipartite graph. Recall that a *biclique* is a complete bipartite graph; a *biclique cover* is a collection $C = \{B_1, \ldots, B_k\}$ of bicliques that cover all the edges of B, i.e., any edge $e \in E$ is in some biclique $B_i \in C$.

In the biclique cover (BC) problem we are given a bipartite graph B, and we need to find for B a biclique cover of minimum cardinality. The BC problem is known to be hard to approximation within factor n^ϵ, for some $\epsilon > 0$, where $n = |V_1 \cup V_2|$ is the size of the graph ([3, 1]).

The BC_d problem is the special case of BC in which the bipartite graph B is bounded-degree in one side; that is, the degree of any vertex $v \in V_1$ is at most d, where $d > 1$ is some constant. The BC_d problem arises e.g. in optimizing the sizes of address tables on the Internet.

Open Problem: Is the BC_d problem NP-hard? (hard to approximate?)

Known Results: BC_d can be approximated within factor d, by a greedy algorithm, and within factor $O(\log d)$ when $|V_1|/|V_2| \leq r$, and $r \geq 1$ is some fixed constant [2].

References

[1] C. Lund and M. Yannakakis. On the hardness of approximating minimization problems. *J. of the ACM*, 41:960-981, 1994.

[2] H. Shachnai. On Bounded-degree biclique covers. Manuscript, 2003.

[3] H.U. Simon. On approximate solutions for combinatorial optimization problems. *SIAM J. on Discrete Math.* **3**:294–310, 1990.

Seffi Naor

0.6 Metric Labeling

Motivated by certain classification problems that arise in computer vision and related fields, Kleinberg and Tardos recently introduced the metric labeling problem. In a typical classification problem, one wishes to assign labels to a set of objects to optimize some measure of the quality of the labeling. The metric labeling problem captures a broad range of classification problems where the quality of a labeling depends on the pairwise relations between the underlying set of objects. More precisely, the task is to classify a set V of n objects by assigning to each object a label from a set L of labels. The pairwise relationships between the objects are represented by a weighted undirected graph $G = (V, E)$, where $w(u, v)$ represents the strength of the relationship between u and v. The objective is to find a labeling, a function $f : V \to L$, that maps objects to labels, where the cost of f, denoted by $Q(f)$, has two components.

- For each $u \in V$, there is a non-negative assignment cost $c(u, i)$ to label u with i. This cost reflects the relative likelihood of assigning labels to u.
- For each pair of objects u and v, the edge weight $w(u, v)$ measures the strength of their relationship. This models the assumption that strongly related objects should be assigned labels that are *close*. This is modeled in the objective function by the term $w(u, v) \cdot d(f(u), f(v))$ where $d(\cdot, \cdot)$ is a metric on the labels L.

Thus

$$Q(f) = \sum_{u \in V} c(u, f(u)) + \sum_{(u,v) \in E} w(u, v) \cdot d(f(u), f(v))$$

and the goal is to find a labeling of minimum cost. We remark that if the distance function d is not a metric, then determining whether a graph can be colored by k colors is a special case of the labeling problem.

Metric labeling has rich connections to some well known problems in combinatorial optimization. It is related to the *quadratic assignment* problem, an extensively studied problem in Operations Research. A special case of metric labeling is the *0-extension* problem, studied by Karzanov. There are no assignment costs in this problem, however, the graph contains a set of terminals, $t_1, \ldots, t_k$, where the label of terminal t_i is fixed in advance to i, and the non-terminals are free to be assigned to any of the labels. As in the metric labeling problem, a metric is defined on the set of labels. Karzanov showed that certain special cases (special metrics) of the 0-extension problem can be solved optimally in polynomial time. Clearly, the 0-extension problem generalizes the well-studied *multi-way cut* problem in which the metric on the label set is the uniform metric.

Kleinberg and Tardos obtained an $O(\log k \log \log k)$ approximation for the general metric labeling problem, where k denotes the number of labels in L, using the probabilistic tree approximations technique. They also gave a 2-approximation for the uniform metric using a linear programming formulation. Chekuri et al gave a natural linear programming

formulation for the general metric labeling problem and proved that the integrality gap of the formulation is $O(\log k \log\log k)$ for general metrics and 2 for the uniform metric.

A natural and interesting question is whether there exists a constant factor approximation algorithm for the metric labeling and 0-extension problems. This is particularly interesting given the rich connection of these problems to other well-studied optimization problems. We note that the best lower bound on the approximability of these problems is only MAX SNP hardness which follows from the MAX SNP hardness of the multi-way cut problem. Understanding the integrality gap of the linear programming formulation mentioned above is a very promising direction. We note that a solution to this linear program has a very interesting *geometric* interpretation. It defines an embedding of the graph in a k-dimensional simplex, however, the distance between points in the simplex is defined by a special metric and not by the (standard) ℓ_1 metric.

There are many special cases of the metric labeling problem that are interesting in their own right from both theoretical and applications point of view. One such special case is the truncated linear metric, where the distance between two intensities i and j is given by $d(i,j) = \min(M, |i-j|)$. This special case is motivated by its direct applications to two problems in computer vision, namely *image restoration* and *visual correspondence*. For the truncated linear metric a factor of $2+\sqrt{2} \simeq 3.414$ can be shown via the above linear programming relaxation. It would be very interesting to study the approximability of the truncated quadratic distance function for which very little is currently known.

0.7 Priority Steiner Tree Problem

In the next telecommunication age it will be possible to support new multimedia applications in a global environment and design new services on flexible platforms without upgrading the physical infrastructure. This requires new network architectures capable of offering transport and computation services to communication applications with stringent quality-of-service (QoS) requirements. A key issue is the provision of network resources so as to meet these requirements. The multicast backbone of the internet (Mbone) is increasingly used for broadcasting live audio and video in digital form all over the world. However, heterogeneity is an enduring characteristic of the Internet creating difficulties in the transmission of real-time multimedia data across groups. Heterogeneity originates, e.g., from the wide range of network transmission rates, varying across many orders of magnitude, and from the vast differences in computing power. Members of a group (receivers) may vary significantly in their characteristics, e.g., bandwidth availability or computing power. This means that a source would be required to transmit in a way that matches the most constrained receiver. Instead, it would be advantageous to send data to multiple receivers at heterogeneous rates and in a way that matches the capability of each individual receiver.

Motivated by these considerations, Charikar et al. studied the following optimization problem, called the *priority Steiner tree problem*. We are given an edge weighted graph representing the network, a multicast root r and a subset of the vertices, the terminals (clients). Each terminal has a priority level, an integer from 1 to K (1 being highest priority and K being lowest), and each edge is labeled with a priority as well. The priority

models the level of service required by a client. The objective is to find a minimum cost tree that connects r to the terminals such that each terminal has a path to the root with every edge on the path having priority at least that of the terminal in question. Thus, each terminal is guaranteed to receive service along the path that meets its requirements. This problem is a clear generalization of the well known Steiner tree problem.

Charikar et al. gave a $\min\{O(\log |X|), 2K\}$-approximation algorithm, where X is the set of terminals. A $2K$-approximation follows easily by finding a separate Steiner tree for each priority level. The logarithmic approximation is obtained by adapting the greedy online Steiner tree algorithm to the priority Steiner tree problem. The terminals are greedily added to the current tree in decreasing order of priority. However, the only hardness of approximation result known is the one from the Steiner tree problem. We note that the priority Steiner problem is also a special case of the directed Steiner problem. This can be seen by replicating the graph K times, where copy i contains edges that have priority at least as high as i. For each vertex v and i, $1 < i \leq K$, there is a zero-cost directed edge from vertex v in copy i to vertex v in copy $i-1$. The root is vertex r in copy 1.

A major open question is whether the priority Steiner tree problem is approximable to within a constant factor. The natural cut relaxation has recently shown to have an $\Omega(\log K)$ integrality gap. A strengthened flow formulation is the following: flow is allowed to reach the root only on paths with edges that have non-decreasing priorities. The integrality gap for the cut formulation does not seem to apply to the flow formulation.

0.8 Network Design: Orientation Constraints

An *orientation constraint* on a pair of nodes u and v states that a feasible solution may include at most one of the arcs (u, v) and (v, u). Orientation constraints arise in many network design problems, since link/edge resources such as fiber, are commonly unidirectional (i.e. they support traffic in only one of the two possible directions at a given time). We are interested in studying the following problem: given a crossing supermodular requirement function and a set of orientation constraints, find a subgraph with a minimum orientation cost that satisfies the requirement function as well as the orientation constraints (the cost of an orientation is defined to be the sum of the costs of the orientations of the edges.)

The cost function associated with the orientation may in general be *asymmetric*, i.e., the cost of orienting an edge $e = uv$ from v to u is different from orienting it from u to v. Asymmetric costs may arise in many network routing problems. For instance, consider the setting where traffic demand is being incrementally introduced in an existing network. Load balancing constraints may favor forcing traffic in opposite directions between a given pair of switches. Hence, when routing new demands, costs on the directed links may increase proportionately to the amount of existing traffic. Asymmetric costs may also arise in network planning due to assorted line termination equipment; these are the costs associated with terminating the two ends of a link.

A closely related problem is the min-cost orientation problem where we are given an undirected graph and the goal is to find a minimum cost orientation that satisfies certain connectivity requirements. In this case, it is known that for strongly connected orienta-

tions, a good characterization for this problem follows from the classical min-max theorem of Lucchesi and Younger. The orientation problem for general crossing supermodular requirement functions can also be solved in polynomial time via a reduction to submodular flows. However, these ideas do not seem to extend in any direct manner to our design problems with orientation constraints. Informally speaking, we can view network design with orientation constraints as a two-phase problem: finding a subgraph of an undirected graph, and orienting its edges so as to satisfy the cut constraints. While each of the subgraph selection problem and the orientation problem is well-understood for crossing supermodular functions, not much seems to be known for design problems that combine together these two constraints.

Perhaps, the most basic design problem with orientation constraints is that of finding a subgraph that admits a minimum cost strong orientation. This problem generalizes two well known NP-hard problems. If the orientation cost function is symmetric, then the problem reduces to finding a minimum cost 2-edge connected subgraph of G. On the other hand, if there are no orientation constraints, then the problem reduces to finding a minimum cost strongly connected subgraph of a directed graph. We note that for both problems, 2-approximation algorithms are known. In a recent work, Khanna et al. and obtained a 4-approximation algorithm.

An interesting open problem in this context is k-strong connectivity with orientation constraints for which no non-trivial bounds are known. Unfortunately, the approach used for strong connectivity completely breaks down as we move to 2-strong connectivity. A possible approach for this problem is to use ideas from the work of Jain (that identifies and rounds iteratively a large component in a basic feasible solution). In this case, it would require proving, for example, that in every basic feasible solution there exists an orientation constraint (on a pair of vertices u and v), where the fractions attached to the arcs (u, v) and (v, u) add up to at least a constant. Thus, in each iteration, our algorithm would then tighten such a constraint and require that the fractions add up to 1. At the end, all orientation constraints would be tight, and then the problem can be solved in polynomial time.

Gordon Wilfong

Two edge coloring problems.

0.9 Edge coloring dynamic bipartite multi-graphs

We are given two sets of nodes A and B and we are told that edges will appear and disappear over time, all edges will have one node in A and one node in B and the degree at any node will never exceed Δ. As edges appear, they must be assigned a color that differs from the colors of all existing edges adjacent to each endpoint of the new edge. That is, the goal is to maintain a proper edge coloring. The question is: How many colors are required to do this?

More formally, let $\delta_E(v)$ be the degree of node v given a set E of edges. Given a set N of nodes and a maximum degree Δ define a set of edges E to be *valid* if and only if

1. for any $\{u, v\} \in E$, $u, v \in N$;
2. for any $v \in N$, $\delta_E(v) \leq \Delta$.

Let the *edge sequence* $\mathcal{E} = (E_1, E_2, \ldots)$ be a sequence of valid edge sets E_i. We define a dynamic graph $\mathcal{G}(N, \Delta, \mathcal{E})$ to be the sequence of graphs $(G_1, G_2 \ldots)$ such that $G_t = (N, E_t)$. We assume that $\mathcal{G}$ starts as just the set N of nodes and define $E_0 = \emptyset$ to be the initial set of edges.

A coloring C assigns a color $C(e)$ to each edge e. We define $\mathcal{C} = (C_1, C_2 \ldots)$, a sequence of colorings, to be a *proper coloring* of $\mathcal{G}$ if for any i

1. C_i is a proper coloring of G_i;
2. for any e such that $e \in E_{i-1}$ and $e \in E_i$, $C_{i-1}(e) = C_i(e)$.

The following theorems are what we know about this problem.

Theorem 0.1 *For any edge coloring algorithm, there is a dynamic bipartite graph* $\mathcal{G}(A, B, \Delta, \mathcal{E})$ *with* $\max(|A|, |B|) = n$ *where* $n = (\frac{1}{4} + \mathrm{o}(1))\Delta^2$ *such that the edge coloring algorithm must use at least* $2\Delta - 1$ *colors to color* $\mathcal{G}$.

Theorem 0.2 *If* $\mathcal{G}(A, B, \Delta, \mathcal{E})$ *has* $|A| = |B| = 2$ *then it can be edge colored with* $3\Delta/2$ *colors.*

Theorem 0.3 *If* $\mathcal{G}(A, B, \Delta, \mathcal{E})$ *has* $|A| = |B| = 3$*, then it can be edge colored with* $15\Delta/8$ *colors.*

Theorem 0.4 *For any edge coloring algorithm, there exists a dynamic bipartite graph* $\mathcal{G}(A, B, \Delta, \mathcal{E})$ *where* $|A| = |B| = 3$ *such that the edge coloring algorithm must use at least* $7\Delta/4$ *colors.*

Theorem 0.5 *For any edge coloring algorithm, any $\epsilon > 0$ and $\Delta > 1/2\epsilon$, there exists a dynamic bipartite graph with fewer than $1/\epsilon^2$ nodes on each side that requires the algorithm to use more than $2(1-\epsilon)\Delta$ colors.*

Thus the question as to how many colors are needed to edge color a dynamic bipartite graph having k nodes on either side where k lies somewhere between 3 and about Δ^2 is wide open. For instance, how many colors are needed if $k = \Delta$? How about $k = 4$?

(This problem comes from joint work with Haxell, Rasala and Winkler.)

0.10 Edge coloring bipartite multi-hypergraphs

Let $G = (A \cup B, H)$ be a multi-hypergraph where A and B are sets of nodes and H is a collection of hyperedges where each $h \in H$ consists of exactly 3 nodes from $A \cup B$ and either h has two nodes from A or two nodes from B. Suppose the maximum degree of any node in G is Δ (that is, each node is an element of at most Δ hyperedges). An edge coloring of G is an assignment of a color to each hyperedge so that no hyperedges with at least one node in common are colored the same color. Then the question is: How many colors are required to edge color G?

Of course, this question can be generalized to any sized hyperedges rather than just those with 3 nodes and other combinations of how many nodes in a hyperedge are from particular subsets of the nodes.

As far as we know, the only things known about this problem are the trivial upper bound of $3\Delta - 2$ and a simple 2Δ lower bound construction.

(This problem comes from joint work with Haxell and Winkler.)

Mansoor Alicherry

0.11 Optimal cost chromatic partition (OCCP)

OCCP problem for a general graph can be defined as follows: Given a graph $G = (V, E)$ with n nodes and a sequence of coloring costs $(k_1, k_2, \ldots, k_n)$, find a proper coloring $C(v) \in \{1, 2, \ldots, n\}$ of each node $v \in V$ such that the total coloring costs $\sum_{v=1}^{n} k_{C(v)}$ are minimum. This problem is a generalization of *chromatic sum* problem.

This problem is NP-hard for a general graph and solvable in linear time for trees. For interval graphs it can be solved optimally in polynomial time if there are only two different values for the coloring costs. However, if there are at least four different values for the coloring costs, then the problem is NP-hard.

Problem: Is OCCP problem, for interval graphs, with only three different values for the coloring costs in P?

George Karakostas

0.12 The single-source unsplittable flow problem

Statement without edge costs: Let $G = (V, E)$ be a directed network with positive edge capacities u_e, a source s and k commodities with terminals t_i and positive demands $d_i, i = 1...k$. We assume that $u_e \geq d_{max} := max d_i$ for all edges, and that a vertex may contain any number of terminals. We are looking for a routing in which each commodity i flows through a single path from s to t_i so that all the demands are satisfied and the total flow through any edge is not more than the edge capacity.

Statement with edge costs: Same as before but now a cost per unit of flow is associated with each edge, and we want to find an unsplittable routing with the minimum cost.

It may not be possible to find any unsplittable flow that satisfies the edge capacities. In this case, we can ask for the smallest $a \geq 1$ (a is the congestion) such that if we multiply all capacities by a, then we can route all demands unsplittably. In case we are not allowed to change the capacities, we can ask for the minimum number of routing rounds needed to satisfy all demands unsplittably.

Known results: The unsplittable flow problem was introduced by Kleinberg. He observed that well-known NP-complete problems like PARTITION, 3-PARTITION and BIN PACKING can be cast as unsplittable flow problems. This fact not only shows that unsplittable flow problems are NP-complete, but that the combination of routing and bin packing characteristics makes these problems particularly hard to solve as well.

For the version without costs, Dinitz, Garg and Goemans (FOCS'98) give a 2-approximation for the minimum congestion (more precisely, each capacity is violated by at most d_{max}) or routing in 5 rounds. For the version with costs, Kleinberg (FOCS'96) gave the first constant factor bicriteria approximation algorithm with (congestion,cost)-approximation factor equal to $(6 + 2\sqrt{5}.3 + 2\sqrt{5})$. This was improved to (3,2) by Klein and Kolliopoulos (FOCS'97), and to (3,1) by Skutella. The main open problem regarding unsplittable flows is whether one can improve this factor (for example, can it be reduced to (2,1), maybe by combining ideas from the works in both the cost and costless versions).

Observations and progress: Many of the participants consider the problem of how to find an unsplittable flow in the network, with capacities $f_a + d_{max}$, whose cost is no more than that of the original (split) flow f. Leonid's modified proof (of a sightly stronger result) was also presented and attempts were made to turn it into a polytime algorithm.

Dinitz, Garg and Goemans prove their result via augmentations along special types of cycles. Vetta and Shepherd showed that graphs without such cycles actually form trees (this was done actually as an attack on the confluent flow problem - see later). We give a sketch of this argument.

Consider any *b-flow* vector f for our problem. By this, we mean that for each node v, the net-flow out of v is b_v the net flow into r is $\sum_{v \neq r} b_v$. We define the *congestion* of v under f, to be the total flow through the node v, i.e., $c(v) = f(\delta^-(v))$. The congestion of

f, denoted $\Phi(f)$ is $\max\{c(v) : v \in V - 2\}$. Clearly, the minimum confluence is at least the minimum confluence of a b-flow to r. In the following we prove:

We consider an acyclic graph $D = (V, A)$. Call a directed path P in D an *I-path* if each of its internal nodes has in-degree exactly one. We call a cycle C (in the undirected sense) *dingo* (Dinitz, Garg, Goemans) if it can be written as $C = P_1, Q_1, P_2, \ldots, P_l, Q_l$ where for each $i > 1$, P_i is an I-path. In addition, P_1 would be an I-path but for a single internal node called the *black sheep*.

In the following, for a digraph D, a *hanging arborescence* is an arborescence T with a leaf v such that $T - v$ is a connected component of $D - v$. We obtain the following structure on acylic digraphs with no dingo cycles.

Claim 0.1 *Let D be a connected acylic digraph. Then either D has a dingo cycle, or a hanging arborescence.*

Proof: Suppose that D is a minimal counterexample and so none of the two structures exist. Suppose there is a cut arc $a = (u, v)$ of D, where v lies in a nontrivial component of $D - a$. Suppose this is not the case, and let D' be obtained by deleting all nodes in v's component except for v. Then clearly no node of D' has out-degree one either. Moreover, any dingo cycle in D' would also have been dingo in D. Thus there is no such cycle, and hence by minimality, there is a hanging arborescence at some node $x \in D'$. But this was then also such an arborescence for D itself. We may henceforth assume that any cut arc (u, v) has the property that v is a leaf. That is D is obtained by hanging some leaves (u, v) from nodes u in some 2-edge-connected digraph D'.

Now basically apply the dingo argument. We grow a cycle, alternately finding an I-path, and then an arbitrary path. Start at a source and greedily grow a path avoiding leaf arcs until we reach a node x of in-degree at least 2. If we cannot do so, we would choose an arc $a = (u, v)$ at some point where v is only adjacent to some pendant leaves. But then the arc a would contradict our earlier assertion on cut arcs.

¿From the node x we now greedily grow a directed path into x, until we back up into some source node y. One then checks that this procedure can always be applied. Moreover, since there is no node of out-degree 1, the node y may be used to start growing a new I-path. Let $P_1, Q_1, P_2, \ldots$ be the paths constructed. Eventually, we repeat a visit to a previous node. There are two cases. Suppose first that in growing some P_l we revisit a previous node w. Note that w cannot be an internal node of some P_i since any such node has in-degree 1. ALso, w cannot be the beginning of an i-path, since any such node is a source. Thus w lies on some Q_i and is not the last node of Q_i. One sees that the cycle identified is then a dingo cycle.

Second, suppose that we revisit a node, again called w, for the first time while growing a Q_l. First if w lies on some P_i but is not the last such node, then starting at w and following $P_i, Q_i, \ldots$ forms a dingo cycle. Otherwise, w lies on some Q_i. Then let z be the last node of that Q_i. Then a dingo cycle is obtained by starting at z and following $P_{i+1}, Q_{i+1} \ldots$ until we reach Q_l and follow the path to w, and then following Q_i to z.

□

Adrian Vetta

0.13 Confluent Flow Problem

We consider a class of multi-commodity flow problems, the *confluent flow* problem. Take a network $G = (V, A)$ and a set of source-sink pairs $(s_1, t_1), \ldots, (s_k, t_k)$. The goal in a generic multi-commodity flow problem is to route r_i units of flow from s_i to t_i. In addition, we wish this flow to have the minimum possible cost. A confluent flow is a specific type of multi-commodity flow that is constrained to use at most one path between any pair of vertices u and v. Confluent flows are now generating widespread interest. This attention is primarily based on the following distinct reasons. Firstly, confluent flows are of very practical importance: destination based routing utilised by the internet produces confluent flows. Secondly, from a mathematical view-point confluent flows correspond to elegant combinatorial structures worthy of study in their own right.

In internet based applications the cost of a flow is (partially) dependent upon the congestion in the network. Therefore the ability to design good algorithms that produce low congestion confluent flows is vital. This task, though, is rather difficult. Consider for example, the specific case in which all the sinks are identical. In such an instance, the task of finding a confluent flow that minimises the maximum congestion on any link corresponds to the graph theoretic problem of finding a rooted arborescence in which the size of the largest subtree is minimised. It was shown by Chen, Rajaraman and Sundaram (STOC 2003) that this problem is MAXSNP-hard, i.e. no polynomially time algorithm can find a solution that is guaranteed to be arbitrarily close to the optimal solution. They also show that that problem is approximable to within a factor $\tilde{O}(\sqrt{n})$ of optimal where n is the number of vertices in the network. Thus the resultant gap between the lower and upper bounds of approximability was large.

During the meeting, Adrian Vetta improved the inapproximability bound significantly from $\Omega(1)$ to $\Omega(\log n)$. Chekuri, Shepherd, Vetta, tried to match this with an $O(\log(n))$ approximation algorithm, using the structural result for graphs with no dingo cycles (see the section on unsplittable flows). One can show that augmenting along a dingo cycle does not increase a node's congestion by more than its offered demand (i.e., 1 in the uniform case). (One may also show a $\log(n)$-approximation if the underlying graph is indeed a tree.) Chekuri also showed that if the support graph of the flow is a layered graph with d levels, then an $O(d)$ approximation can be obtained.

Dan Bienstock

0.14 Combinatorial Algorithms for Short-Path-Decomposable Flows

Given a directed graph $G = (V, A)$ and distinct vertices $s, t \in V$ the s-t maximum flow problem is well known. We are interested in finding flows that do use short paths - that is paths with some bound K on the number of edges in the path. We call such a flow K-hop restricted. A couple of questions are of interest. First, is there a combinatorial algorithm to find the maximum K-hop restricted flow? Second, given a K-hop restricted flow in the compact form (only edge flows are given), can we decompose the flow into K-hop restricted flow paths using a combinatorial algorithm? Bienstock has described an algorithm for these using a time-expanded graph.

Randeep Bhatia

0.15 Problems based on shortest path routing

In all these problems we assume a network $G = (V, E)$ with capacities $u(e)$ and cost $c(e)$ for all links $e \in E$. In our model all routing happens on shortest paths (determined based on link costs). Specifically at any given time a flow of f unit between nodes s and t is routed on a shortest path between s and t in the graph $G' \subseteq G$, where G' contains only those links of G that can accommodate an additional f units of flow. We assume that the shortest paths are unique and that the shortest path computation algorithm implements some kind of tie breaking rule.

Maximum concurrent flow problem: In this problem we are given a demand matrix $D = \{d(s,t)\}$ and we want to compute the largest λ such that it is possible to route at least $\lambda d(s,t)$ aggregate flow between all pairs of nodes s, t. More formally we are interested in finding the largest λ and a sequence of tuples $(s_i, t_i, f_i), i = 1, 2, ...,$ such that f_i amount of flow can be routed between s_i and t_i in the graph G_i at the i-th iteration ($G_1 = G$, G_2 is the graph obtained by routing f_1 units of flow between s_1 and t_1 in G_1 and so on). In addition

$$\sum_{i|s_i=s,t_i=t} f_i \geq \lambda d(s,t).$$

Minimum congestion with minimum link cost changes: In this problem the links capacities can be exceeded. Thus a link e can accommodate more than $u(e)$ unit of flow. Note that in this case all the flow between any pair of nodes s and t is routed on a single (shortest cost) path. The congestion of a link e with flow $f(e)$ is $f(e)/u(e)$. We are given a demand matrix D and a congestion bound θ. The objective is to find a minimum set of links (and their new link costs) such that by changing the link cost of these links the maximum congestion on any link for routing the demand matrix D is at most θ.

Minimum congestion with minimum link additions: In this problem the links capacities can be exceeded. There is a special sink node s and every other node i wishes to route n_i units of flow to node s. Note that these flows will be routed on a shortest path tree rooted at node s. Given a congestion bound θ the objective is to add a minimum set of links, each incident on node s and each with capacity B such that with a suitable choice of costs for the newly added links the maximum congestion on any link is at most θ. This problem is as hard as set cover. Does there exist a $O(\log n)$ approximation algorithm?

Note added Sept 12: *We have learned recently from Seffi Naor that these results have been settled by a team of several collaborators from MIT, Technion, Bell Labs and IBM Research.*

Bruce Shepherd

0.16 Packing Dijoins and Feedback Arc Sets

The following problem is open as far as we know. Given a simple planar digraph, partition the arcs into 3 feedback arc sets. (A feedback arc set is a set of arcs whose deletion results in an acyclic digraph.)

This is an easily stated version of a more general conjecture due to Woodall. Our goal is to discuss several weighted versions of Woodall's conjecture. For a digraph $D = (V, A)$ and set $S \subseteq V$, we denote by $\delta^+(S)$ the set of arcs with tail in S and head in $V - S$. (We define $\delta^-(S)$ similarly.) A *directed cut* in a digraph D is a set of arcs of the form $\delta^+(S)$ such that $\delta^-(S) = \emptyset$. We often refer to S as a *shore* of the cut. A *dijoin* is a set A' of arcs such that $A' \cap \delta^+(S) \neq \emptyset$ for each directed cut $\delta^+(S)$. Woodall's conjecture states that any directed graph contains k disjoint dijoins if each directed cut contains at least k arcs.

Note that if D is planar, then a set of arcs is a dijoin if and only if they form a feedback arc set in the planar dual of D. Thus for $k = 3$, Woodall's conjecture coincides with the opening conjecture.

A more bold conjecture was made by Edmonds and Giles. Namely, suppose that the arcs of D have integer nonnegative weights w_a. Let k now denote the minimum weight of a directed cut. A *w-packing of dijoins* is a collection of dijoins such that each arc is contained at most w_a dijoins in the collection. Schrijver disproved this conjecture by exhibiting a digraph and $0, 1$ vector w such that $k = 2$ but for which there existed no pair of disjoint dijoins amongst the arcsx $\{a : w_a = 1\}$. Nevertheless, we may have approximate packing results in the vein of Erdoš-Posa or Gallai-Younger. Namely, we ask whether there is an increasing function $f()$ such that each such digraph contains a w-packing of $f(k)$ dijoins. Indeed, this is likely tied to resolving even the following simple question. Does there exist a constant C, such that if D is a digraph whose minimum w-weight directed cut is at least C, then D contains a w-packing of 2 dijoins.

In the positive direction, Shepherd-Vetta showed that there is always a half-integral packing of $\frac{k}{2}$ disjoint dijoins. They also reduce the question of determining the existence of two disjoint dijoins (in the weighted setting) to that of finding an integral point in a certain weakly submodular flow polyhedron. The problem is also related to a question of Bang-Jensen and Jensen who asked whether large enough directed-in and -out connectivity from a node v, is sufficient to guarantee the existence of 4 disjoint arborescences rooted at v: two directed in, and two directed out.

References

[1] D. Bienstock. Personal communication, August 2003.

[2] G. Călinescu, H. Karloff, and Y. Rabani. An improved approximation algorithm for multiway cut. Journal of Computer and System Sciences, 60:564-574, 2000.

[3] C. Chekuri, S. Guha, and J. Naor. Approximating Steiner k-cuts. In *Proceedings of ICALP*, 2003.

[4] E. Dahlhaus, D. S. Johnson, C. H. Papadimitriou, P. D. Seymour, and M. Yannakakis. The complexity of multiterminal cuts. SIAM J. on Computing, 23:864–894, 1994.

[5] J. Edmonds. Optimum branchings. J. Res. Nat. Bur. Standards, B71:233-240, 1967.

[6] A. Freund and H. Karloff. A lower bound of $8/(7 + 1/(k - 1))$ on the integrality ratio of the Călinescu-Karloff-Rabani relaxation for multiway cut. *Information Processing Letters*, 75(1-2): 43-50, 2000.

[7] M. Goemans and D. Williamson. A general approximation technique for constrained forest problems. SIAM J. on Computing, 24:296–317, 1995.

[8] O. Goldschmidt and D. Hochbaum. Polynomial algorithm for the k-cut problem. Mathematics of Operations Research, 19:24–37, 1994 .

[9] D. Karger and C. Stein. A new approach to the minimum cut problem. Journal of the ACM, 43:601–640, 1996.

[10] D. Karger, P. Klein, C. Stein, M. Thorup, and N. Young. Rounding algorithms for a geometric embedding of minimum multiway cut. In *Proceedings of the 29th ACN Symposium on Theory of Computing*, pp. 668-678, 1999.

[11] J. Naor and Y. Rabani. Approximating k-cuts. In *Proceedings of the 12th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 26–27, 2001.

[12] R. Ravi and A. Sinha. Approximating k-cuts via Network Strength. In *Proceedings of the 13th Annual ACM-SIAM Symposium on Discrete Algorithms*, pp. 621–622, 2002.

[13] H. Saran and V.V. Vazirani. Finding k-cuts within twice the optimal. SIAM J. on Computing, 24:101–108, 1995.

[14] V. Vazirani. Approximation Algorithms. Springer, 2001.